Proof-of-experience: empowering proof-of-work protocol with miner previous work

Samuel Masseport, Benoit Darties, Rodolphe Giroudeau, Jorick Lartigau

Proof of Experience: empowering Proof of Work protocol with miner previous work

Samuel Masseport[1,2], Benoît Darties[2] , Rodolphe Giroudeau[2], and Jorick Lartigau[1]
[1] Pikcio SAS, Montpellier, France
{samuel.masseport, jorick.lartigau}@pikcio.com
[2] LIRMM, University of Montpellier, CNRS, Montpellier, France
{samuel.masseport, benoit.darties, rodolphe.giroudeau}@lirmm.fr

***Abstract*— Even after 12 years of services, Bitcoin remains the leading electronic money system and the most influential player in the crypto-currency market. For each block, the underlying distributed proof-of-work consensus, which requires a great deal of computing power (used by miners) to solve a difficult mathematical problem, rewards the first miner who solves the problem and shares the solution with the rest of the network. Two main problems regarding this consensus can be easily highlighted: (1) the computational power spent by the miners is wasteful since only one of them will be rewarded, and (2) miners with a much lower computational capacity can almost never solve the mathematical problem, and thus cannot earn a reward. Also the historical players in the system see their reward greatly diminished if a new player with a high computational capacity suddenly enters the game.**

This paper proposes a substantial amendment to the existing Bitcoin consensus by using the wasted computation to adjust the difficulty of each miner that did provide computation effort overtime. Therefore, anytime a miner justifies of its work toward the consensus hence the network, without reaching the solution, then its difficulty will decrease accordingly for next blocks.

The impact of this adjustment is threefold: (1) It allows to "recycle" part of the energy previously spent in the computation of a previous block, which makes the system less energy consuming as a whole; (2) It also reduces the calculation time of a block and thus speeds up the validation of transactions; (3) It increases the loyalty of the actors by favoring the old miners who have been investing in the system for a long time, compared to a new entrant with a high computing power, who would monopolize all the rewards otherwise.

I. Introduction

In November 2008, the first fully decentralized crypto-currency, bitcoin, was introduced by an unidentified person or group acting under the pseudonym Satoshi Nakamoto [23]. Although this article is not the starting point of the Blockchain technology - proposed by Haber and Stronetta in 1991 [13] - it is however the one that will make it known to the general public by using it as an immutable and decentralized storage medium for transactions operated in Bitcoin [28]. New transactions are included by adding blocks to the blockchain. Each block is added by a single node determined by the system, recognized by consensus by all other nodes in the network as being able to add a new block of transactions to the blockchain. To date, Bitcoin is the most widely adopted and trusted economic system built on a peer-to-peer network, enabling online payments directly from one party to another without a single fiduciary entity.

The Bitcoin consensus is based on a proof-of-work (PoW) protocol[12], [14] that protects the network from denial-of-service (DoS) and double-spending attacks. In a PoW protocol, nodes (called "miners") compete to add a block to the current chain by trying to solve a difficult asymmetric mathematical problem.[1] The Bitcoin blockchain rewards the miner for adding a block to the current chain.

However, the main problem with PoW is that it requires a significant amount of computing power. Currently, Bitcoin's power consumption is estimated to be about 77.78 TWh per year[2] (about the annual energy consumption of Chile), and much of this power consumption is considered wasted by miners trying to solve the mathematical problem without success.

Some previous work has attempted to reuse or reduce the power consumption of bitcoin, by proposing alternative consensus mechanisms [2].

Permacoin [21], based on the proof of Retrievability [15] is a blockchain technology that requires users to commit both computational power and storage for the purpose of creating a platform for distributed archival data storage. Other blockchain solutions such as Foldingcoin [7], Golem[8] or Gridcoin[9] offer crypto-currencies whose mathematical problems are coupled with complex scientific calculations. Thus, miners are rewarded according to the amount of scientific calculations they have performed. Other recent works, such as Proof of Deep Learning [6] use the computing power of miners on deep learning algorithms.

Other previous work proposes rewarding the loyalty of miners. Initially, Proof-of-Stake [24] (PoS) was proposed to replace the mechanisms based on computing power with another, based on the active use of one's capital. It makes use of the blockchain asset in question, for example tezzies (XTZ) or EOS (EOS) on the Tezos [10] and EOS smart-contract platforms. Because of this, it is not possible to simply plug in and start mining like with PoW. You have to buy it or earn it in some way. Thus, blockchains in PoS are always subject to fundraising in order to make an initial distribution of the native asset. The holders of the asset in question have the possibility to put in sequestration a part or even the totality of their capital with the objective to

[1] An asymmetric mathematical problem is a mathematical problem that is difficult to solve but whose solution is easy to verify.

[2] https://digiconomist.net/bitcoin-energy-consumption

participate directly in the validation of the blocks or else to vote for an actor who will do it for them according to the implementations of this system. For example, Peercoin [17] relies on proof-of-stake based consensus, amending it with a randomization concept called "age of the coin". The miner who is allowed to add a block is selected based on a number derived from the product of the number of coins by the number of days the coins have been held. Thus, older and larger coin sets have a higher probability of signing and closing the next block. Coins that have not been spent for at least 30 days can compete for the next block. Peercoin incentivizes nodes to stack coins and reward them based on their seniority. The result is a model where nodes are encouraged to stack their coins rather than spend them.

The Proof-of-Activity [3] (PoAc) mechanism combines the two mechanisms PoW and PoS. Initially, PoAc starts with the mining process, in which different miners try to outbid each other with more computing power to find a new block. Once this new block is found, which unlike PoW only contains a header and the miner's wallet address, the system switches to the proof-of-stake mechanism. Based on the header, a new random group of validators is selected to validate the new block. In this mechanism, the more stakes a validator has, the more chances he has to validate the new block. Once all validators have signed the new block, it is given the status of a complete block. Since the PoA system marries PoW and PoS, it draws some criticism for its partial use of both. It still takes too much energy to mine blocks during the PoW phase, and coin hoarders are still more likely to get on the sign-up list and accumulate more coin rewards. In [4] authors propose a flexible version of PoAc with tunable parameter for PoW and PoS. TwinsCoin [11] is an example of public blockchain crypto-currencies using a consensus based on both PoW and PoS.

Proof-of-Elapsed-Time [1] (PoET) is another blockchain network consensus mechanism proposed by Intel in 2016 as another alternative to PoW that prevents high resource utilization and energy consumption; it keeps the process more efficient by following a fair lottery system that spreads the chances of winning equally across network participants, giving every node the same chance. Under PoET, each participating node must wait for a randomly chosen period; the first to complete its waiting time wins the right to add the new block. Each node in the blockchain network generates a random wait time and sleeps for that specified duration. PoET is often used on permissioned blockchain [25] networks to decide the mining rights or block winners on the network. Permissioned blockchains are networks that require any potential participant to identify themselves before they are allowed to join.

In this paper, we propose a new consensus protocol, namely Proof of Experience (PoE), to reward miners based on the amount of computational work they have provided over a given period of time (e.g., over the past few months). Proof of Experience is an extension of the Bitcoin consensus in which a miner's unsuccessful computational work can be turned into experience. The more experience a miner accumulates over time, the less computational capacity they will need to add incoming blocks.

Currently, in the PoW protocol, a miner with a larger computational capacity than others is more likely to add blocks. Therefore, let's consider a crypto-currency use case in which a miner with a small computational capacity who is working for the long-term stability of the blockchain would like to reap the benefits of its loyalty.

Indeed, early backers or emerging blockchain services or technologies are the backbone of the infrastructure and its sustainability as a potential success. Once the blockchain network has reached a momentum that attracts larger miners with heavy computing capabilities, the miners who provided early support may find themselves drowned in the mass and excluded from the reward process. When this happens, many blockchain technologies lose their early adopters and believers, who often help launch commercial achievements and service reliability.

The loss of these adopters and believers can have a significant negative impact on this technology. Being the first to believe in a crypto-currency, they are also the first to use it and be the standard bearers for it. On the other hand, new entrants are more opportunistic, interested in the gain that a technology can bring them, more than the technology itself. These new entrants can quickly turn away from a technology if a more profitable competing technology emerges. To allow a crypto-currency to be sustainable over time, it is therefore essential to be able to retain its early adopters and believers.

PoE provides a protocol to reduce the influence of large miners/pools of miners that will rise through the attraction of potential reward. It also balances the loyalty of the miners against the computing power.

Another motivation for the PoE protocol is to boost network distribution by attracting miners with smaller computational capabilities, which allows location and identities to be spread over a large centralized pool. A miner with reduced computational power can still compete using its loyalty, without having to waste all the effort involved in distributed consensus, leading to network security.

Although other consensus systems, such as PoET [1], [5], achieve the "one CPU one vote" goal originally proposed in Nakamoto's Bitcoin paper, these are mostly restricted to permissioned or private blockchains, which is not public accessible and require relative trust of the nodes involved. Moreover, they put the same probability of winning on both the old lazy miners and those who actively participate in block calculations.

In the PoE protocol, the previous computational work can be derived into experience, in order to benefit from a reduced difficulty when solving the mathematical challenge of incoming blocks. Indeed, the experience reduces the difficulty of the nonce that must be solved to legitimize the addition of a block, thus increasing the probability of rewards for the miner.

Compared to other solutions and consensus proposed, PoE can reconcile many advantages that are not found or

partially in other solutions. It only requires a minor but clever adaptation of PoW, and thus is completely suitable for both public and private blockchains. It does not take into consideration the amount of pieces available, already spent, used, or burned as it is the case in other consensus systems, such as PoS, PoAc, PoU [18] or PoB [16]. These systems tend to favor the richest players and/or those who make the most transactions, and can induce a sense of inequity, as adopters and believers are not necessarily those who own the most assets in the system. These systems tend to favor the richest players and/or those who make the most transactions, and can induce a sense of inequity, as adopters and believers are not necessarily those who own the most assets in the system. Moreover, consensus systems based on quantities of money induce changes in the behavior of the actors (saving, spending) in order to maximize the gains, which seems to us to be counterproductive in such a system. In this sense PoE allows to free itself from these behaviors biases by the very fact of its conception. PoE is also more robust to different attacks well identified on these blockchain networks (double spending, cloning attack).

II. Protocol

In the PoE protocol, we propose a modification of the existing Bitcoin consensus. This modification is not a complete redesign of the Bitcoin protocol, but an ingenious adaptation of PoW to make it more equitable and less energy intensive. The PoE model uses the same transaction model (based on unspent transaction output, UTXO), the same signature system (based on the elliptic curve[22]) and the same asymmetric mathematical problem as Bitcoin's PoW. This problem consists of finding a 32-bit number called *nonce* such that the concatenation of the block header (i.e. the block without transaction list) and the nonce produces a hash less than or equal to the target. The target is a 256-bit number, shared by all Bitcoin miners, that varies every 2016 blocks depending on the overall computational capacity of the network. Every time the overall computing capacity increases, the target decreases, just as when the overall computing capacity decreases, the target increases. The difficulty of finding a valid nonce depends on the value of the target. The lower the target is, the higher the difficulty is.

Note that in such a protocol, the SHA-256 hash function returning a 256-bit number is used. Also note that in this document we distinguish between PoW (i.e., the nonce problem) and PoE (the proof that a miner worked for the previous blocks). The term *work* means that a miner provides computational power while trying to solve the nonce problem.

In PoE, the miner's experience is represented by the work it has done previously. The more work a miner does for the network is the lower the difficulty level will be. In fact, there are two types of targets in PoE: (1) the global target, which is the maximum value of a target, shared by all miners (corresponding to the Bitcoin target), and (2) the local target, specific to each miner: when a miner mines for the very first time, his target is equal to the global target. However, by working for the network, each miner can reduce the global target and calculate his own local target. For each block, each miner calculates his own local target based on the previous PoW he reached - even if the computed nonce produces a hash higher than the global target.

In PoE, as in PoW, each miner tries to compute the next block b_i to earn the reward. To compute the next block b_i, a miner m must perform the following operations:

1) **Target computation:** m computes its own target t_m based on the global target and its previous work.
2) **Header construction:** m constructs the PoW of the block and computes a nonce such that the hash of the PoW concatenated to the nonce returns a hash less than or equal to t_m.
3) **Transactions selection:** m adds transactions from its transaction pool to the current block.
4) **Block sharing:** m computes the final hash of the block and shares the block over the network.

In order to stabilize the network, as in the case of Bitcoin, miners always work on the longest chain. When a new block is sealed, it is communicated to all other active nodes.

In this section, we present the block structure used in the PoE protocol. Thus, it will be demonstrated how to compute the target and to disseminate it on the network. The last parts of this section are devoted to the validation of the blocks and to the study of special cases (e.g., fork).

A. Block structure

For a better understanding, it is encouraged to assimilate the following Table I which describes each entry within a block.

In the Bitcoin blockchain, the header contains all the information of the block (but not the transaction stack of the block but rather its root Merkle [19], [20]). The challenge of mining is to find a nonce such that the hash of the header will be less than or equal to the target that is broadcast by the network. With PoE, there is a clear distinction between the PoW itself and the header, i.e., the header contains the PoW but the proof is not executed on the entire header. Compared to Bitcoin blocks, a PoE block contains a pool of nonces, the ID of the miner computing the block and a local target. The other entries are not changed (see figure 1). These changes are needed to facilitate the calculation of the local target. With this change, a miner m_1 can prove his experience on a set of blocks $B = \{b_1, \ldots, b_n\}$ to a miner m_2 by simply revealing his public key or ID, and the ordered sequence of nonces computed for each block (corresponding to the nonces pool in Figure I). The block indexes of B are fixed and known to the two exchanging miners so that m_2 can reconstruct the n headers, recompute the hashes to finally determine the local target of m_1.

The global target and nonce pool are not included in the header since lightweight nodes (i.e., nodes that download block headers only to validate the authenticity of transactions) do not require these fields to validate transactions.

Field	Description
Version	Version of the block.
Previous Block Hash	The block hash of the block that this block is being built on top of. This is what "chains" the blocks together.
Merkle Root	All of the transactions in this block, hashed together. Basically provides a single-line summary of all the transactions in this block.
Timestamp	When a miner is trying to mine he add the time at which the block header is being hashed.
Bits	A 32-bit number corresponding to a shortened version of the Target. The target is a 256-bit number that miners must find a hash such that this hash is lower than or equal to the target. The conversion between bit and target is described in Appendix.
Nonce	The field that miners change in order to try and get a hash of the block header (a block hash) that is below the target.
Miner ID	The ID of the miner that create the block.
Pool of nonces	In this field, the miner write the previous nonces that he have find for some previous blocks (see Section II-B for more details).

TABLE I

FIELDS CONTAINED IN BLOCKS.

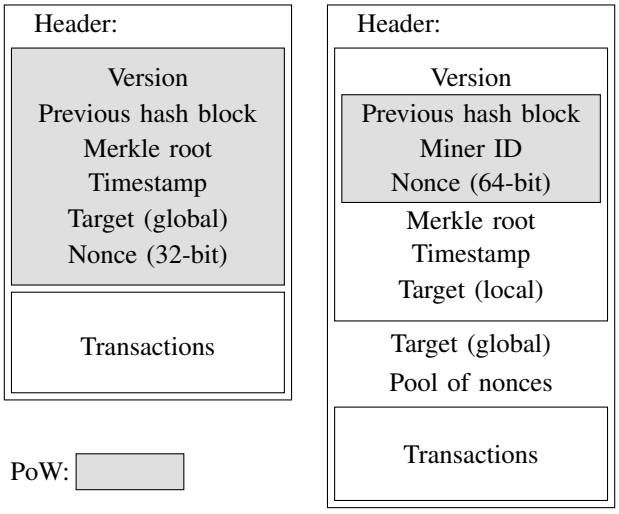

Fig. 1. Structure of Bitcoin blocks on the left compared to the structure of PoE blocks on the right. Gray rectangles correspond to fields where the PoW is executed.

B. Local target

This part is devoted to the computation of the local target for a given miner based on its previous work. In order to add a block, a miner must find a nonce such that concatenating this nonce with the hash of the previous block and its identifier or unique address (derived from its public key) produces a hash less than or equal to its local target. In addition, the miner must prove his previous experience by adding a pool of nonces corresponding to his previous work. Each nonce of this pool corresponds to a computational effort on a previous block of the blockchain, effort produced during the process of adding said block. The local target of this pool varies according to the work that the miner has previously performed. When a miner m shares a block, the others miners must be able to recalculate that the hash is valid from the given nonce and to verify its local target - that is to say, to recheck that it corresponds to the previous computational effort: the pool of nonces contained in the blocks thus corresponds to an ordered set of nonces that the miners use to reconstruct the PoW of the previous blocks and thus recalculate the local target of m according to the hashes obtained.

Ideally, the calculation of the block target should consider all blocks since genesis. However, the computation time might be too long and the size of the nonces pool (contained in the blocks) too large. In order to reduce the computation time and to limit the size of the blocks, we do not consider all the blocks but only a significant part (i.e. a sample) of the last previous blocks. The considered sample can be described by a function, and is considered here as a parameter of our protocol, as explained in the following.

1) Sample from previous blocks: The sample of previous blocks corresponds to a set of blocks that a miner considers to calculate the target of a block (both to exploit and to verify a block in the consensus phase). Let $S = \{s_1, s_2, \ldots, s_n\}$ (with $s_1 < s_2 < \cdots < s_n$) be a set of positive integers, and let m_1 be a miner writing b_i the i^{th} block of the chain. To compute m_1's local target for b_i, a miner m_2 (possibly m_1) considers m_1's work on blocks b_{i-s_j}, $\forall s_j \in S$. Using such process a miner can estimate the work of a miner on s_n blocks by checking only n blocks.

The elements and size of this set can be fixed and shared by all the miners of the network or calculated dynamically (in this case, the function which calculates it must be deterministic and known by all the miners). An example of a fixed set can be $S = \{x \mid x \text{ is a odd number} \wedge x \in [\![1, 100]\!]\}$. In this case the size and the elements of S are fixed and known by all actors of the network, thus, miners can compute the local target of a block without computing S. Now if we consider the set $S = \{x \mid x = i - 100k, \forall k \in [\![1, i/100]\!]\}$, with i the number of the current block, the elements and size of this set are not fixed. Then,Then, each time a miner computes the local target, it must compute the previous set.

Considering the i^{th} block b_i, the size of the set (i.e. $|S|$) corresponds to the number of blocks evaluated to compute the local target of b_i, and the position of the current block minus the largest value of the set (i.e. $i - s_n$) corresponds

to the number of the oldest evaluated block. Thus, the miner who starts mining will get the full benefit of his experience after working continuously for s_n blocks. We present the set S chosen in subsection III-A.

2) Local target computation: This section is dedicated to the local target computation function. This function is used by miners both to compute their own target, and to compute others during the process of validating or invalidating a block when they receive one. As said before, each miner has a different target for the same block b_i depending on the previous work provided. The more previous work a miner provides, the easier the work required for the next block will be. In fact, there is an overall target in the network (as in Bitcoin) that decreases for each miner based on his previous work on the chain. In this section, we define the function determining the target of a block b_i for a given miner m as a function of his work on previous blocks in the given set $B = \{b_j \mid j = i - s_k, s_k \in S\}$ with an arbitrary set S of positive integers. The inputs/outputs of this function are defined as follows:

> ***Input:*** given a set $S = \{s_1, s_2, \ldots, s_n\}$ (with $s_1 < s_2 < \cdots < s_n$) of positive integer, a set of nonces $N = \{x_1, x_2, \ldots, x_n\}$ (i.e. a pool of nonces), the miner ID of the miner m and a current block b_i.
>
> ***Output:*** a 256-bit number corresponding to the local target t_m of the miner m for the block b_i.

Note that the set of nonces N is the set of best nonces found by the miner m (i.e. the nonces producing the lowest hash). The set of nonces is ordered: the nonce x_k corresponds to the nonce found for the PoW of block b_{i-s_k}.

To compute the local target, a miner m' (possibly $m' = m$) must recreate and then verity the work of m for each block of B. Thus, $\forall s_k \in S$, m' creates the PoW of m_s for block b_{i-s_j} by concatenating the following elements:

- the hash of block b_{i-s_j-1} as previous hash block,
- the miner ID of m (contained in the block b_i),
- the nonce x_{s_j} in the nonces pool of block b_i.

Thus with only $|B|(= n)$ hashes, m' can compute and evaluate the work done by m for all blocks of B. This work evaluation adjusts m's local target for the block b_i and then this block becomes easier to add.

Note that if the target computation function only uses the evaluation of the work provided by the previous blocks, the chain may be vulnerable to a double-spending attack. In fact, over time, the miner with the highest computational resource on the network may end up with the largest target (i.e., the easiest PoW to find). If this miner is malicious, it can create a local fork, then create a chain that is longer than the rest of the network and force its way onto that chain. In such a case, a double-spending attack is carried out, so that the transactions in progress since the fork are lost and all the work (thus the experience) of the other miners too.

In order to protect the network against the double spending attack, another parameter is used to calculate the target. This parameter is the number of blocks a miner has added recently. For example, if a miner adds eight blocks in the last ten blocks, the function that calculates the target must decrease that miner's local target for the next few blocks so that the blocks become harder to add. Any miner can control this effectively by dynamically storing the miner ID of the last few blocks.

The most critical aspect of PoE consensus is how to adjust the local target based on a miner's work and the number of recently added blocks? There are many functions to adjust this, and each will have a direct impact on the behavior of the network. The subsection III-B presents a model used to calculate the target, which we then tested through simulations.

C. Creation of a nonces pool

Let $S = \{s_1, s_2, \ldots, s_n\}$ (with $s_1 < s_2 < \cdots < s_n$) be the positive integers considered to create the sample of blocks.

Property 1: Consider that a miner m locally stores the nonce that produces the smallest hash for each block between the current block b_i and the block b_{i-s_n}. Then a miner can create its nonce pool in constant time $O(|S|)$.

Proof: Consider that m stores its nonces in an array of size s_n. For each j^{th} block b_j, m stores the corresponding nonce in the indexed array j modulo s_n. Thus, $foreverys_k \in S$, m accesses the nonce of block b_{i-s_k} by accessing the index of array $i - s_k$ modulo s_n. Then m creates its nonce pool for block $i+1$ by accessing $|S|$ times to its array. ■

D. Block Verification

Let m' be a miner which receives a block b_i constructed by miner m. In order to validate b_i, m' must check the following conditions:

- m has correctly computed its local target for block b_i (based on its experience that the nonces pool of the block produces).
- The hash of the PoW is less than or equal to the local target.
- The transactions are valid (this step is exactly the same for the Bitcoin blockchain).

If all these conditions are validated, then the block is valid and m' adds it to its chain.

The only change between a Bitcoin block check with or without the PoE protocol is the calculation of the local target of b_i for m_s. The time for this verification depends on the chosen function.

E. Special cases

This part is devoted to the behavior of PoE in two particular cases of the blockchain: the beginning of the chain and the fork.

1) The beginning of the chain: Considering the set of samples $S = \{s_1, s_2, \ldots, s_n\}$ (with $s_1 < s_2 < \cdots < s_n$), we consider the beginning of the chain as the chain before the ${s_n}^{th}$. Then at the beginning of the chain, there are blocks needed to calculate the local target which does not exist. In this case, the calculation of the target is executed only with

the existing blocks. Thus, miners working from the first block will take full advantage of their experience on the $(s_n + 1)^{th}$ block of the chain.

2) Fork: As with bitcoin, a fork can occur in the PoE protocol when two or more miners find and share a block at nearly the same time. The fork is resolved when blocks are added to one of the chains, and then when that chain becomes longer than the others. Thus, considering that miners always work on the longest chain, they will work for that chain and blocks that are not in the longest chain are dropped by the network. These blocks are called "orphan blocks". To the network, orphan blocks do not exist, so the work the miner has done for them cannot be verified. For this reason, the work provided for the orphan blocks cannot be used for the calculation of the local targets and unfortunately this work is lost for the miners.

III. Discussions

This section presents a relevant sample and target function test scenario to illustrate PoE. Note that this section only analyzes the theoretical aspects of our protocol, and requires more extensive testing for full validation. The protocol proposed in section II is currently under development.

A. Sample of blocks

In order to simplify the calculation of the local target, the set S is fixed. This set is known by all the actors of the network. This set is defined in such a way that miners who do not work regularly will be strongly penalized. For example, let us consider the set $S = \{10k \mid \forall k \in [\![1, n]\!]\}$ (with n an arbitrary integer) which does not penalize miners, even if they only work 1 time out of 10, as long as they work regularly. A set that we find particularly interesting to consider as a sample of blocks to check for PoE, is a set built on a Golomb ruler [26], [27] : a *Golomb ruler* is a set of marks located at integer positions along an imaginary ruler, such that the distance between any two marks is pairwise distinct. The *order* of a Golomb ruler is the number of its marks, and the *length* of a ruler is the largest distance between any two pairs of marks. A set of positive integers $A = \{a_1, a_2, \ldots, a_n\}$ (with $a_1 < a_2 < \cdots < a_n$) is a Golomb ruler of order n and length $a_n - a_1$ if and only if:

$\forall i, j, k, l \in \{1, 2, \ldots, n\}\ a_i - a_j = a_k - a_l \wedge i \neq j \iff (i = k) \wedge (j = l)$.

Property 2: Considering that $S = \{s_1, s_2, \ldots, s_n\}$ (with $s_1 < s_2 < \cdots < s_n$) is a Golomb ruler, for any pair of blocks (b_i, b_j) (with $i < j$), there exists at most one block b_c that uses both b_i and b_j to calculate its target.

Proof: Consider that b_i and b_j are used to compute the local target for block b_c. Then there exist $s_k, s_l \in S$ such that $c - s_l = i$ and $c - s_k = j$. By contradiction, let us consider that b_i and b_j are used again to compute the local target of the block b_{c+d} (with $d > 0$). Then there are $s_{k'}, s_{l'} \in S$ such that $c + d - s_{l'} = i$ and $c + d - s_{k'} = j$. Thus $s_k = s_{k'} - d$ and $s_l = s_{l'} - d$, then we have $d = s_{k'} - s_k = s_{l'} - s_l$ (with $s_k \neq s_{k'}$) such that $s_k \neq s_l$ and $s_{k'} \neq s_{l'}$ then S is not a Golomb ruler. ■

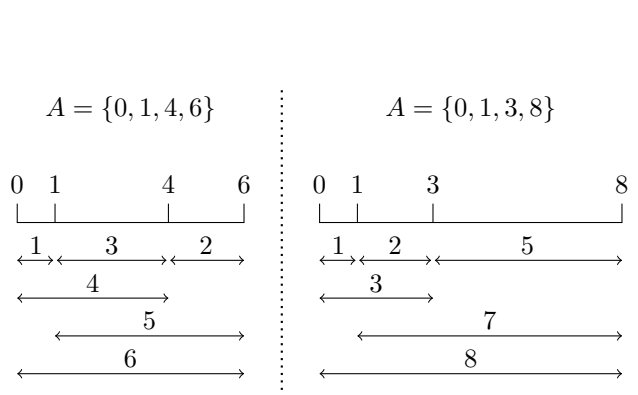

Fig. 2. Two examples of Golomb rulers, on the left a Golomb ruler of order 4 and length 6, on the right a Golomb ruler of order 4 and length 8. Note that the Golomb ruler on the left is optimal (i.e. its length is minimal considering its order) but it is not necessary for us to have an optimal ruler.

Consider that S is a Golomb ruler, according to Property 2, for any pair of blocks (b_i, b_j) (with $i \neq j$), there exists at most one block that uses b_i and b_j to compute its target. By this property, a miner who tries to cheat by mining only a few blocks is penalized because a pair of mined blocks is used at most once in computing the target.

Considering the i^{th} block b_i, the order of the Golomb ruler (i.e. $|S|$) corresponds to the number of blocks evaluated to compute the target of b_i, and the current block position minus the biggest value of the ruler (i.e. $i - s_n$) corresponds to the number of the oldest block evaluated.

For the following tests, let us consider the following Golomb ruler of order 212 and length 41 911 (see Figure 3) that can be found on Lloyd Miller's web page[3] (this ruler is exposed in Appendix). Considering the Bitcoin blockchain in which a block is added every 10 minutes on average, this ruler allow to check the work of a miner for the last 292 days (approximately 10 months) by processing only 212 blocks (i.e. executing 212 hashes).

The size of a block is then increased due to the pool of nonces, the public address or ID of the miner and the local target it contains. The average block size in bitcoin in February 2020 is 1.105 MB[4]. The nonces pool consists of 212 nonces, each corresponding to a 32-bit number (4 bytes). The size of the nonces pool is therefore 844 bytes. The miner ID is a number of 256 bits (32 bytes) and the bits corresponding to the local target are a number of 32 bits (4 bytes). Then PoE, with a S set of size 212, increases the block size by 880 bytes (a 0.08% increase). If we consider the Bitcoin chain from the genesis block to March 8, 2020 (currently 267 GB), this change increases the chain size by about 547 MB (about a 0.2% increase). If we only consider the headers (for lightweight nodes), we only consider the miner ID. Thus, by the same arguments, the size of the chain for lightweight nodes corresponds to 69.70 MB (currently 49.78 MB for the Bitcoin chain). This is an increase of 14%.

As we can see from Figure 3, the marks in the proposed Golomb rule are distributed equitably and uniformly. However, for equivalent work, a miner who has worked on older

[3] `http://www3.telus.net/millerlf/g3-records.html`
[4] urlhttps://ycharts.com/indicators/bitcoin_average_block_size

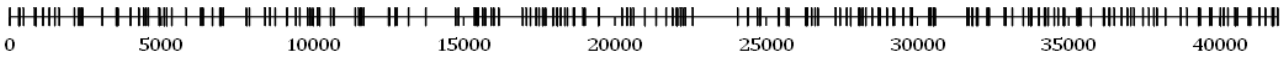

Fig. 3. Golomb ruler of order 212 and length 41 911.

blocks must gain more experience than those who have only worked on recent blocks. It may be interesting to test the Golomb rule with a weighting on the corresponding set. With this weighting, the older a block is, the more impact said work will have on the target calculation.

B. Target computation function

As said before, there are many functions that can be used to compute the local target t_m of a block b_i for a miner m. The function proposed here is based on a global target t_g such that the work on previous blocks increases and the number of recently added blocks decreases. In the following, a target is represented by its greatest power of 2. Let the global target equal $t_g = 2^G$, with G an integer such that $1 < G < 255$. Note that G is recomputed by all miners every 2016 blocks to adjust the global target so that one block is produced every 10 minutes (as in Bitcoin). The proposed local target calculation function is defined as follows:

$$t_m = 2^G \times 2^{f(\alpha)} \times 2^{-g(\beta)} = 2^{G+f(\alpha)-g(\beta)},$$

with α the evaluation of the work of m and β the number of blocks recently added by m. The functions f and g are two monotonuous increasing functions. They are used to adjust the target of m according to α and β.

Let b_{i+1}, the $(i+1)^{th}$ block of the chain, be the block on which we compute the local target. The value of α is an estimate of the work done by m for all the blocks $B = \{b_{i-s_j} \mid s_j \in S\}$ (with b_i the i^{th} block of the chain). In order to evaluate this work, for all $b_k \in B$ blocks, a miner recreates the PoW and recomputes the h_k hash found by m (as described in subsection II-B) to create the set of H_m hashes. Note that finding a hash less than 2^n is half the work of finding a hash less than 2^{n-1}. Then, for each hash $h_k \in H_m$, we count the number of 0 (named $h_k^{|0|}$) before the first 1 (from left to right) and add $2^{h_k^{|0|}}$ to α. Thus, α corresponds to a sum of $2^{h_k^{|0|}}$ of all $h_k \in H$. Then:

$$\alpha = \sum_{h_k \in H_m} 2^{h_k^{|0|}}.$$

The value of β is based on the number of blocks added by m in the last s_n blocks (i.e. in the last 10 months). The value of each block is balanced, i.e. it is weighted according to its seniority. The newer the block, the heavier it is. For example, if b_j is the j^{th} block, we can weight the blocks with the function $w(b_j) = max\{0, s_n - (j-i)\}$ (with i the number of the current block). With this weighting, a block starts with a weight of s_n and decreases one after another for each block to reach 0 after s_n blocks. Consider the set $B_m = \{\text{blocks added by } m \text{ in the last } s_n \text{ blocks}\}$, the variable β corresponds to the sum of the weights of the blocks added by m :

$$\beta = \sum_{b_j \in B_m} w(b_j).$$

Property 3: Consider that a miner m locally and dynamically stores the ID of the miner who added the block between the current block b_i and the block b_{i-s_n}. The miner m computes $B_{m'}$ the set of blocks added by m' in the last s_n blocks (possibly $m = m'$) in constant time $O(s_n)$.

Proof: Consider that m stores the miner ID of the last blocks in an array of size s_n. For each j^{th} block b_j, m stores the corresponding miner ID in the array indexed j modulo s_n. Thus, by traversing this array, m creates the set $B_{m'}$ in s_n comparisons between the miner ID of m' and the miner IDs contained in the array. ■

Further tests are needed to determine the functions f and g even if they are monotonically increasing functions.

IV. Tests

In this section, we propose some local target computation functions to reduce the influence of a miner joining the network with high computing power. To test the effectiveness of our functions, we implemented a simplified version of the Bitcoin protocol. This version contains no transactions, no asymmetric block signatures and low - the objective is mainly to simulate and prove the feasibility of our approach, not to propose a finalized version of PoE - . Tests are performed with the Python 3 language[5] and running on four Raspberry Pi 2 model B[6], one laptop (Asus R511L[7]) and one graphic card (MSI Geforce GTX 1080 Ti GAMING X 11G[8]), all connected on local network by a switch (Cisco Catalyst 2950[9]).

In order to evaluate the influence of a miner joining the network with significant computing power, we defined a test protocol as follows :

- All but the most powerful miner have been working since the beginning of the network to accumulate experience.
- After s_n blocks, the miners reach the maximum experience they can have with their computing power (with s_n the maximum value of the used sample S). The

[5] https://www.python.org/download/releases/3.0/

[6] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[7] https://www.topachat.com/pages/detail2_cat_est_ordinateurs_puis_rubrique_est_wport_puis_ref_est_in10090896.html#fiche-technique

[8] https://www.msi.com/Graphics-Card/GeForce-GTX-1080-Ti-gaming-x-11G/Specification

[9] https://www.cisco.com/c/dam/global/fr_fr/assets/documents/pdfs/datasheet/switching/Cat2950_fr_v3.pdf

sixth miner (with the highest computing power) starts working for the network.

The objective of this protocol is to observe the evolution of the percentage of blocks that each miner adds to the chain, and to measure the influence of the sixth miner on the network when he joins the system.

Before presenting our functions and results, we need to determine the computational power that each of our miners has on the network. This power is expressed as a percentage of the overall network power. To do this, we use the following protocol : the six miners work at the same time on the network with a consensus without proof of experience (only with proof of work). The percentage of blocks added by each miner corresponds to the percentage of its power compared to the total power of the network. The results we obtain are presented in the following table: II.

Device	Percentage of recorded blocks
Laptop	34,2
Graphic card	52,2
Raspberry 1	3,5
Raspberry 2	3,4
Raspberry 3	3,1
Raspberry 4	3,6

TABLE II

PERCENTAGE OF RECORDED BLOCKS BY EACH MINER OF THE NETWORK WITHOUT PROOF OF EXPERIENCE (ON 5370 BLOCKS).

As we observe, the miner with the most computing power is the one with the graphics card. In the following tests, the graphics card will be our sixth miner and will enter the network from the $s_n + 1^{\text{th}}$ block.

As said before, many functions can be used to calculate the local target. However, we are looking for a function with which the incoming miner has very little influence on the network and reaches its full potential only after working on s_n blocks (in our protocol, starting from the $2s_n^{\text{th}}$ block). For obvious reasons of time (more than twenty months of testing for each function with the $S_2 12$ sample and ten minutes per block), we use an average time of one minute between each block and the $S_{61} = [s_i + 1 \;\; s_i \in A_{61}]$ sample (with $s_1 < s_2 <, \ldots, < s_{61}$), a sample based on Golomb's rule A_{61} of order 61 and size 3134 :

$A_{61} = \{0, 4, 28, 44, 94, 300, 371, 409, 513, 544, 611, 632, 689, 745, 763, 775, 804, 836, 984, 987, 1083, 1105, 1125, 1242, 1251, 1265, 1409, 1442, 1626, 1652, 1662, 1677, 1679, 1749, 1754, 1833, 1928, 1965, 2008, 2118, 2157, 2203, 2211, 2279, 2314, 2476, 2536, 2588, 2601, 2702, 2721, 2766, 2829, 2929, 2935, 2936, 2984, 3018, 3065, 3123, 3134\}$.

Let t_m be the local target of the miner m for the block b_i, the i^{th} of the chain. The first function we present is based on a bonus/malus system. The more a miner has worked, the more bonus he has. The number of malus increases according to the number of blocks the miner has added recently. The function we propose is the following :

- Reconstruct the proofs of work of the set of blocks corresponding to the sample, then compute the set H, the set of hashes of their proofs of work.
- In the last 256 blocks of the chain, for each block b_j added by m, add to the set $Malus$ the value $256 - (b_i - b_j)$. For example, if m added the previous block and another block 102 blocks ago, we add to $Malus$ the value 255 (=256-1) and 154 (=256-102). Thus, each block recorded by a miner costs him a malus, dynamically weighted by the inverse of his age. In other words, the more recently a block was added, the higher its weight.
- Compute $Sum = \sum H + \sum Malus$, the sum of the hashes of H plus the sum of the malus.
- Compute $M = Sum/|S|$. At this point, we computed the average work done by the miner m on the sample S to which we added a malus based on the recently recorded blocks.
- Let T_{max} and T_g be the value of the maximum target of the network and the value of the current global target respectively, compute $t_m = T_g + (T_{max} - M)$. This calculation allows us to invert the average we just calculated so that the more a miner works, the higher his local target is.
- It is important that the computed target is always between T_{max} and T_g, so the function returns T_{max} if $t_m > T_{max}$, T_g if $t_m < T_g$ and t_m otherwise.

With this function, the more a miner works for the network, the higher his local target is. On the other hand, the more recent blocks he adds, the lower his local target is. This malus avoids a runaway situation in which a miner with more computing power than another would have a simpler and simpler target which would decrease the chances of the other miner to add a block.

The behaviors of miners on a network using the previously proposed function are observable in the appendix, Figure 4.

In the 3135 blocks, we observe a certain stability, the laptop registers between 70% and 80% of the blocks of the chain and each raspberry between 5% and 10%. When the graphics card enters the network, it directly reaches its full potential and then the network balances with the same proportions as for the network without proof of experience. This behavior is explained by the fact that the bonus given for the experience of the miners does not compensate for the malus that is given when they add a block. Thus, when the graphics card starts mining, all the miners in the network have a local goal equal to the global goal (exactly as in a network without proof of experience).

To address this problem, we added a dynamic weighting to the blocks based on their age. This new function is the same as the previous one except that for each block corresponding to the element s_i of the sample S its hash is weighted by $\lceil log_{10}(s_i) \rceil$. Thus, the previous block will have a weight of 1, while the block added 2128 blocks ago will have a weight

of 3. The behavior of the miners with this new function is presented in appendix, Figure 5.

In this new function, we observe at first that the proportion of blocks recorded by the laptop increases more and more at the expense of raspberrys. This behavior can be explained by a too low malus compared to the bonus given to the laptop, it is a runaway as described above. However, with this function, when the graphics card joins the network (at block 3135), it has very little power on the network, it must wait for more than 1500 blocks before starting to add blocks and the full experience (3135 blocks) to reach its full potential.

From our point of view, the behavior of the miners resulting from the second function is "better" than the first one in spite of the runaway observed in the first 3135 blocks, and allows to show that PoE allows to favor the miners working for the network for a long time compared to the new entrants, despite a more important computing power. The objective will be to test other functions in order to remove this runaway.

V. Conclusion and perspectives

This paper proposes an empowering evolution of the Proof of Work protocol in which the miner's loyalty is preserved by rewarding and capitalizing on his previous work, thus his experience.

Experiments conducted on real devices allow us to demonstrate the feasibility of the Proof-of-Experience mechanism: we were able to privilege the historical nodes of the system by "reusing" their energy by adapting the difficulty level, and we were able to make the network less energy consuming. We have shown through these experiments that the addition of a new node with a large computational power does not immediately lead to a monopolization of all the rewards of the network: the historical actors manage to maintain a percentage of rewards for a certain time despite this new actor, which reinforces the loyalty of the actors. In addition to being feasible, our protocol is intended to be adaptive: the ability with which old nodes can compete with new entrants depends on a target computational function that can be adjusted according to the desired behavior.

Future work consists of further testing of the proposed sample and the target computational function to adjust these parameters based on the observed results.

We have focused here on the loyalty problem, proposing a consensus system that by design will be less energy intensive and faster - we do not make the associated mathematical problem harder for new players, but easier for old ones - . Another extension work consists in testing other couples of sample/target computation functions in order to compare them with some parameters such as target computation time, block size, verification time, energy saved compared to a POW consensus, and also to get a better behavioral analysis of the network participants. Ideally, we would like to provide a feature for each type of application (e.g., one feature favoring recent work and another more dedicated to past work and history).

Another possible improvement for proof of experience will be to introduce a special type of transaction that can propagate experience from one miner to another. This modification would allow miners to change their public and private key pair to monetize their experience or simply port their legacy to a new account.

References

[1] Marius Paulius Asadauskas, Christian Cachin, and Ignacio Amores Sesar. Poet: an eco-friendly alternative to pow, 2021.

[2] Seyed Mojtaba Hosseini Bamakan, Amirhossein Motavali, and Alireza Babaei Bondarti. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Systems with Applications*, 154:113385, 2020. URL: https://www.sciencedirect.com/science/article/pii/S0957417420302098, doi:https://doi.org/10.1016/j.eswa.2020.113385.

[3] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y. *SIGMETRICS Perform. Eval. Rev.*, 42(3):34–37, dec 2014. doi:10.1145/2695533.2695545.

[4] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y. *SIGMETRICS Perform. Eval. Rev.*, 42(3):34–37, dec 2014. doi:10.1145/2695533.2695545.

[5] Lin Chen, Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, and Weidong Shi. On security analysis of proof-of-elapsed-time (poet). In Paul Spirakis and Philippas Tsigas, editors, *Stabilization, Safety, and Security of Distributed Systems*, pages 282–297. Springer International Publishing, 2017.

[6] Changhao Chenli, Boyang Li, Yiyu Shi, and Taeho Jung. Energy-recycling blockchain with proof-of-deep-learning. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 19–23. IEEE, 2019.

[7] Foldingcoin company. Foldingcoin whitepaper. Technical report, Foldingcoin, 2018. https://foldingcoin.net/index.php?option=com_content&view=article&id=236&Itemid=653.

[8] Golem company. Golem whitepaper. Technical report, Golem, 2016. https://golem.network/crowdfunding/Golemwhitepaper.pdf.

[9] Gridcoin company. Gridcoin whitepaper. Technical report, Gridcoin, 2018. https://gridcoin.us/assets/img/whitepaper.pdf.

[10] Thi Thu Ha Doan and Peter Thiemann. Towards Contract Modules for the Tezos Blockchain. In Bruno Bernardo and Diego Marmsoler, editors, *3rd International Workshop on Formal Methods for Blockchains (FMBC 2021)*, volume 95 of *Open Access Series in Informatics (OASIcs)*, pages 5:1–5:9, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2021/15429, doi:10.4230/OASIcs.FMBC.2021.5.

[11] Tuyet Duong, Alexander Chepurnoy, Lei Fan, and Hong-Sheng Zhou. Twinscoin: A cryptocurrency via proof-of-work and proof-of-stake. In *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*, pages 1–13, 2018.

[12] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.

[13] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3:99–111, 1991.

[14] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Secure Information Networks*, pages 258–272. Springer, 1999.

[15] Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597, 2007.

[16] Kostis Karantias, Aggelos Kiayias, and Dionysis Zindros. Proof-of-burn. In *International conference on financial cryptography and data security*, pages 523–540. Springer, 2020.

[17] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper*, 2012.

[18] Samuel Masseport, Jorick Lartigau, Benoit Darties, and Rodolphe Giroudeau. Proof of usage: User-centric consensus for data provision and exchange. *Annals of Telecommunications*, 75(3):153–162, 2020.

[19] Ralph C Merkle. Method of providing digital signatures, January 5 1982. US Patent 4,309,569.

[20] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[21] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. Permacoin: Repurposing bitcoin work for data preservation. In *2014 IEEE Symposium on Security and Privacy*, pages 475–490. IEEE, 2014.

[22] Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer, 1985.

[23] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[24] Cong T. Nguyen, Dinh Thai Hoang, Diep N. Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, 2019. doi:10.1109/ACCESS.2019.2925010.

[25] Julien Polge, Jérémy Robert, and Yves Le Traon. Permissioned blockchain frameworks in the industry: A comparison. *ICT Express*, 7(2):229–233, 2021. URL: https://www.sciencedirect.com/science/article/pii/S2405959520301909, doi:https://doi.org/10.1016/j.icte.2020.09.002.

[26] JP Robinson. Optimum golomb rulers. *IEEE Transactions on Computers*, 12(C-28):943–944, 1979.

[27] James B Shearer. Some new optimum golomb rulers. *IEEE Transactions on Information Theory*, 36(1):183–184, 1990.

[28] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *Big Data (BigData Congress), 2017 IEEE International Congress on*, pages 557–564. IEEE, 2017.

APPENDIX

Bits is just a shorthand version of the Target. Bits is a 32-bit number (i.e. a 4-byte number), the first byte correspond to an exponent that gives the size of the target in bytes and the other three bytes correspond to the initial three bytes of the target. See the following example for more details:

Bits:

0x 18 06 96 F4

The red byte corresponds to the exponent and the blue bytes correspond to the initial three bytes of the target.

Target:

0x 06 96 F4 00 00 00 00 00 00 00 00 00

The Golomb ruler of order 212 and length 41 911 used as sample for tests:

$S = \{0, 6, 298, 333, 458, 823, 868, 1096, 1305, 1487, 1637, 2136, 2279, 2360, 2413, 3057, 3520, 3542, 3575, 3990, 4268, 4433, 4507, 4578, 4938, 4957, 5104, 5186, 5355, 5834, 6312, 6387, 6699, 6943, 6981, 7048, 7821, 7919, 7937, 8438, 8600, 8774, 9162, 9457, 9586, 9846, 9932, 10021, 10167, 10232, 10240, 10607, 10706, 11420, 11538, 11592, 11615, 11695, 12522, 12556, 12732, 12789, 13179, 13748, 13759, 14736, 14830, 15362, 15430, 15454, 15455, 15481, 15633, 15718, 15914, 16002, 16163, 16953, 17077, 17208, 17397, 17493, 17615, 17631, 17710, 17962, 18088, 18102, 18208, 18347, 18430, 18630, 18662, 18952, 19012, 19458, 19468, 20240, 20403, 20520, 20624, 20633, 20992, 21364, 21693, 21908, 22038, 22059, 22066, 22173, 22251, 22327, 22574, 24059, 24397, 24399, 24715, 24805, 25401, 25421, 25675, 25745, 26316, 26320, 26363, 26504, 26619, 26716, 27299, 27443, 27458, 27663, 27795, 28065, 28101, 28192, 28292, 28506, 28712, 28768, 28999, 29193, 29198, 29235, 29532, 29785, 29825, 30388, 30437, 30524, 30585, 31636, 31649, 31695, 31804, 31816, 31953, 31997, 32308, 32371, 32881, 32922, 33138, 33501, 33672, 33744, 33773, 34013, 34199, 34238, 34322, 34524, 34652, 34829, 34859, 35280, 35342, 35390, 35741, 36152, 36183, 36339, 36356, 36509, 36751, 36941, 37052, 37164, 37455, 37653, 37817, 37919, 38198, 38707, 38915, 39291, 39343, 39662, 39720, 40016, 40124, 40266, 40504, 40507, 40573, 40943, 41007, 41301, 41502, 41728, 41778, 41911\}$.

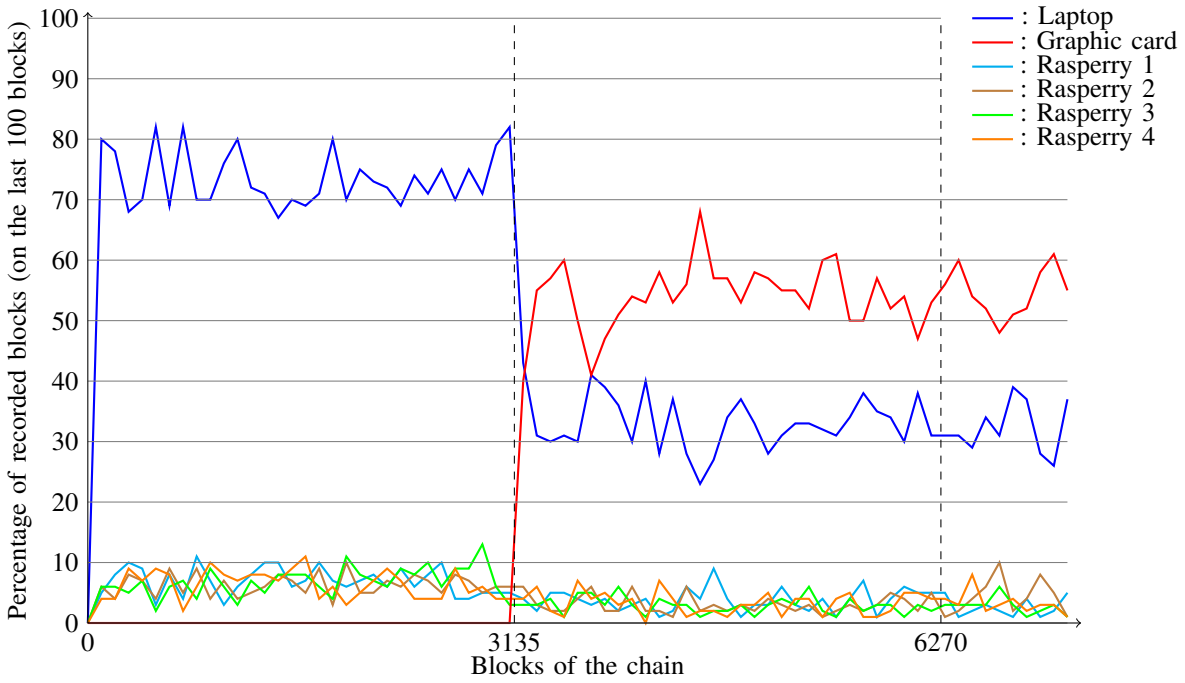

Fig. 4. Average number of blocks recorded by each miner during the creation of the chain (first function).

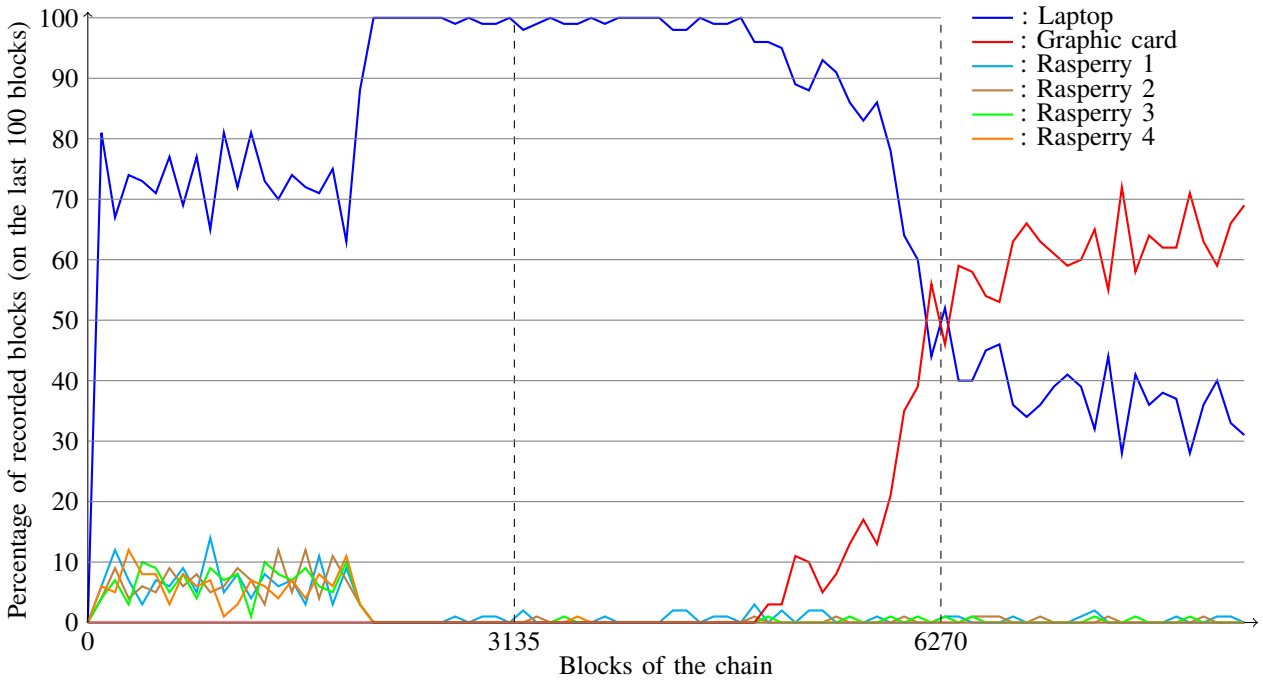

Fig. 5. Average number of blocks recorded by each miner during the creation of the chain (second function).